Jesse Naumanen

SOSIAALISTEN TOIMINNALLISUUKSIEN HYÖDYNTÄMINEN MOBIILISOVELLUKSESSA

Opinnäytetyö
Tietojenkäsittelyn koulutusohjelma

2017

Kaakkois-Suomen ammattikorkeakoulu

Kaakkois-Suomen ammattikorkeakoulu

Tekijä/Tekijät	Tutkinto	Aika
Jesse Naumanen	Tradenomi (AMK)	Marraskuu 2017
Opinnäytetyön nimi Sosiaalisten toiminnallisuuksien hyödyntäminen mobiilisovelluksessa		34 sivua 0 liitesivua
Toimeksiantaja Oy MHG Systems Ab		
Ohjaaja Arto Väätäinen		

Tiivistelmä

Opinnäytetyössä tutkitaan sosiaalisten toiminnallisuuksien hyödyntämistä mobiilisovelluksessa, esitellään sosiaalisien medioiden rajapintoja ja sovelletaan tietoa rakentamalla toimeksiantona mobiilisovellukseen sosiaalisia toiminnallisuuksia. Toimeksiannon tavoitteena on kehittää WittyC-mobiilisovelluksesta sosiaalisempi osana sen pelillistämishanketta lisäämällä sovellukseen ominaisuuksia ja elementtejä, jotka yhdistävät ja houkuttelevat käyttäjiä.

Sosiaalisten toiminnallisuuksien tutkiminen tehdään vertaamalla niitä pelillistämisen keinoihin ja tarkastelemalla sosiaalisten elementtien hyötyjä peleissä ja mobiilisovelluksissa. Sosiaalisten medioiden rajapintoja esitellään yleisesti melko lyhyesti, minkä jälkeen esittely rajataan toimeksiannossa käytettävään Facebookin Graph APIin.

Case-osiossa esitellään toimeksiannossa käytetyt tekniikat ja rakennetut toiminnallisuudet. Lopputuloksena toteutuksessa syntyi sovellukseen mahdollisuus yhdistää Facebook-kirjautuminen Googleen, pistelista sovellusta käyttävien Facebook-kaverien kesken ja sovelluksen sisäinen laskujen ja tulosten jakamisjärjestelmä. Sovellus julkistettiin Google Play Storessa suljettuna valituille betatestaajille.

Asiasanat

sosiaaliset toiminnallisuudet, sosiaalinen media, rajapinta, mobiilisovellus, Graph API

South-Eastern Finland University of Applied Sciences

Author (authors)	Degree	Time
Jesse Naumanen	Bachelor of Business Administration	November 2017

Thesis Title

Utilizing social functionalities in mobile applications

34 pages
0 pages of appendices

Commissioned by

MHG Systems Ltd

Supervisor

Arto Väätäinen

Abstract

This thesis had three objectives: to study how to utilize social functionalities in mobile applications, to introduce the application programming interfaces (API) of different social media and to apply the results to a targeted mobile application. The commission was to make WittyC more social by adding functionalities and elements to it that would connect and attract people.

The study was done by comparing the purpose and means of social functionalities with gamification and by researching the benefits of social elements in applications and games. After introducing social media APIs at a general level, Facebook Graph API was found to be suitable to use for the purpose of the study, and therefore it was focused on.

As a result, three different functionalities were added on WittyC. A friend list was developed to connect different users and to create a foundation for competition. The data for the list is requested from Facebook API every time it's needed, which was why it was necessary to add a possibility to combine Google and Facebook logins. The last functionality was an in-app sharing system for calculations and achievements. The finished version of the application was released in Google Play Store for chosen beta testers.

Keywords

social functionalities, social media, mobile application, Graph API

SISÄLLYS

1 JOHDANTO

Opinnäytetyöni yleisenä tavoitteena on tutkia sosiaalisten toiminnallisuuksien hyödyntämistä mobiilisovelluksissa. Esittelen erilaisten sosiaalisten medioiden rajapintoja ja syvennyn sitten Facebookin Graph API -rajapintaan. Sovellan tietoa toimeksiannossa rakentamalla sovellukseen rajapintoja hyödyntäen erilaisia sosiaalisia toiminnallisuuksia.

Toimeksiantajana toimii MHG Systems Oy. Yritys on ohjelmistosuunnittelija ja -kehittäjä metsä- ja bioenergialiiketoiminnan alalla. Se tarjoaa mobiiliratkaisuja ajantasaiseen metsäomaisuuden, biomassojen, materiaali- ja henkilöstöresurssien hallintaan sekä kaukokartoitustekniikoita ja puukauppaportaalin erilaisille metsäalan toimijoille. Yritys on aines- ja energiapuun hankinta- ja logistiikkaprosessien toiminnanohjausjärjestelmien edelläkävijä, joka osallistuu ahkerasti bioenergia- ja metsäalan tutkimus- ja kehityskärkihankkeisiin.

Toimeksiannossa käytettävä mobiilisovellus WittyC on harjoittelijavoimin kehitetty työkalu hiilijalanjäljen tarkkailuun. Sovelluksella voi laskea käyttäjän hiilijalanjäljen matkailun, jätteiden, lämmityksen ja ruokailun osalta. Sen tarkoituksena on opettaa ja antaa vinkkejä kestävämmästä ja luontoa vähemmän kuluttavista elämäntavoista sekä innostaa ihmisiä kiinnittämään huomiota ympäristöasioihin hauskalla ja sosiaalisella tavalla.

Toimeksiannon aihe on osa keväällä 2017 käynnistettyä projektia, jonka tarkoituksena on lisätä WittyC-sovellukseen pelillisiä ja sosiaalisia elementtejä. Vaikka osuuteni rajautuu sosiaalisiin toiminnallisuuksiin, sivuan työssä asiakokonaisuuden vuoksi myös pelillistämistä.

2 SOSIAALISET TOIMINNALLISUUDET OSANA PELILLISTÄMISTÄ

Tässä luvussa määrittelen käytettävät käsitteet ja esittelen sosiaalisien toiminnallisuuksien hyötyjä sovelluksissa ja peleissä.

2.1 Sosiaaliset toiminnallisuudet ja pelillistäminen

Pelillistämisen ideana on lisätä pelien ominaisuuksia sellaisiin sovelluksiin, joilla ei välttämättä ole kontekstin puolesta mitään yhteistä pelien kanssa. Kuten Haonperä (2013) asian määrittelee, pelillistäminen ei ole liiketoiminnan viihteellistämistä tai sen muuttamista yksinkertaiseksi peliksi tai leikiksi, vaan asiakkaiden, kumppanien ja henkilöstön sitouttamista, motivointia, kouluttamista ja johtamista peleistä tuttuja elementtejä soveltamalla.

Sosiaalisilla toiminnallisuuksilla tarkoitan mitä hyvänsä funktiota tai ominaisuutta, joka jollain tapaa yhdistää ja edistää vuorovaikutusta käyttäjien välillä. Sosiaalisia toiminnallisuuksia voivat olla esimerkiksi keskustelupalstat, pistetaulut, jakotoiminnallisuudet, hyödykkeiden vaihtaminen ja myynti, monipelit tai vaikkapa pelien käyttäjälähtöinen ”modaus”, englanniksi modding eli muokkaaminen.

Sosiaalisten toiminnallisuuksien tarkoituksena on tehdä sovelluksesta houkuttelevampi, lisätä kilpailuhenkeä, kannustaa ihmisiä kokeilemaan sovellusta ja helpottaa sovelluksen leviämistä. Osa sosiaalisista toiminnallisuuksista risteävät pelillistämisen kanssa ja ovat toiminnoltaan samoja, vaikka periaatteet ovat erilaisia. Esimerkiksi klassinen pistetaulu on yleinen peleistä löytyvä toiminnallisuus. Pelillisestä näkökulmasta se näyttää pelaajan edistymisen pisteiden keruussa, kannustaa johonkin päämäärään ja on oleellinen osa pelin koukuttavuutta. Sosiaalisesta näkökulmasta se taas yhdistää eri pelaajia, luo kehyksen kilpailulle ja kenties nimien tai nimimerkkien kautta auttaa osaltaan rakentamaan sovelluksen tai pelin ympärille käyttäjäyhteisöä. Kuten Groh (2012) asian ilmaisee, saavutetut tavoitteet tai onnistumiset ovat paljon mielekkäämpiä, jos ne voidaan jakaa muille.

2.2 Sosiaalisien toiminnallisuuksien hyötyjä

Ihminen on sosiaalinen eläin, joka yleisesti ottaen pyrkii välttämään yksin olemista mahdollisimman paljon. Useimmiten emme halua syödä, nukkua, työskennellä tai leikkiä yksin. Vaikka hetki lukien, liikkuen tai vaikkapa meditoiden omassa rauhassa on mukava yksinolon nautinto, ihmisillä on tapana käyttää enemmän aikaa sosiaalisesti. Sama heijastuu pelisuunnittelun historiassa jo satojen vuosien takaa: valtaosa kehitetyistä peleistä on suunniteltu pelattaviksi muiden kanssa. Yksinpelit olivat hyvin harvinaisia ennen tietokoneaikaa ja nykyään laitteistojen ja yhteyksien parantuessa myös videopeleissä on vuosi vuodelta yhä enemmän monipelaajatoiminnallisuuksia. (Schell 2008, 354.) Siten voidaan ajatella, että sosiaalinen pelaaminen on meille luonnollista ja mielekästä. Mutta mitä sellaista me oikeastaan haemme monipeleistä, mitä yksinpelit eivät tarjoa? Miksi monipelit ovat niin suosittuja? Schell (2008, 355) listaa viisi syytä:

1. Kilpailu

   Monipelit täyttävät useita erilaisia tarpeita. Yhtäaikaisesti ne tarjoavat pohjan, jolla käyttäjät ovat samalla lähtöviivalla ja varteenotettavan vastustajan. Koska vastassa on ihmispelaaja, se mahdollistaa pitkälle kehiteltyjen strategioiden, valintojen ja psykologisten peliliikkeiden kehittelyn. Tämän ansiosta voimme vertailla tulostamme tai taitotasoamme muihin.

2. Yhteistyö

   Kilpailun vastakohtana, yhteistyö sallii meidän harrastaa sellaista toimintaa ja käyttää strategioita, mitkä eivät yksin olisi mahdollisia. Esimerkiksi yksi vastaan yksi jalkapallossa ei ole juurikaan järkeä.

3. Tapaaminen

   On helppoa tavata ihmisiä pelaamisen tai harrastuksen yhteydessä. Se on yhteinen mielenkiinnon kohde, joka yhdistää joukkoa ja tarjoaa hyvän puheenaiheen. On helppoa keskittyä monipeliin ja viettää aikaa yhdessä ilman, että ilmapiiristä tulee kiusallinen.

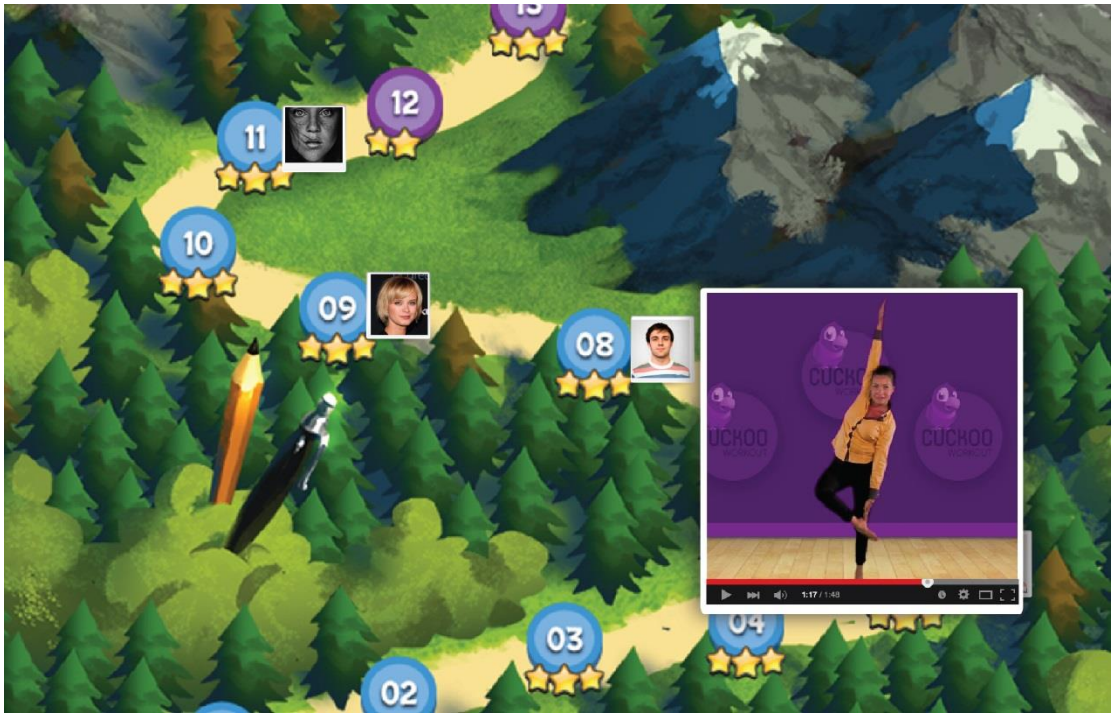
4. Muihin tutustuminen

Tavallisessa kanssakäymisessä ja keskustelussa kuulemme kaveriemme mielipiteitä heille mieluisista tai epämieluisista asioista tai tarinoita heidän ja muiden käyttäytymisestä. Nämä mielipiteet ja tarinat kuitenkin suodattuvat sen näkemyksen mukaan, jonka kaverimme luulevat meidän haluavan nähdä ja kuulla. Sen sijaan pelien ympärillä näemme pieniä kaunistelemattomia vilauksia kaveriemme luonteesta. Pääsemme seuraamaan lähietäisyydeltä, kun kaverimme ratkovat ongelmia ja tekevät valintoja paineen alla. Näemme, päästävätkö he jonkun pälkähästä vai puukottavatko he selkään. Opimme, keneen voi luottaa ja keneen ei. Tunnin pelihetkessä saatamme oppia kavereistamme enemmän kuin vuodessa normaalia kanssakäymistä.

5. Itsetuntemus

   Yksinpelit saattavat haastaa meitä ylittämään rajamme, hahmottamaan mistä pidämme ja missä haluamme kehittyä. Porukalla pelatessa taas koemme, miten käyttäydymme monimutkaisissa sosiaalisissa tilanteissa. Onko meillä tapana antaa kaveriemme voittaa vai murskaammeko heidät armotta? Kenen kanssa lyöttäydymme kimppaan ja miksi? Miten reagoimme häviöön ja miten se eroaa muista? Nämä ovat tärkeitä asioita, lähellä sen ydintä miten näemme itsemme ja miten suhtaudumme muihin.

Sosiaalisten ominaisuuksien logiikkaa peleissä voidaan hyvin hyödyntää myös mobiilisovelluksissa. Kuten pelillistämisessä lisätään sovellukseen pelillisiä elementtejä, myös sosiaalisia toiminnallisuuksia voidaan integroida suhteellisen helposti tuomaan lisäarvoa sovellukseen. Ominaisuudet toimivat parhaiten yhdessä, joten pelillistäessä olisi siten hyvä sisällyttää myös sosiaalisia toiminnallisuuksia kehitettävään sovellukseen, jos ne siihen mitenkään sopivat. Esimerkiksi työpaikoille suunnatussa Cuckooworkout-taukojumppasovelluksessa työntekijät muodostavat tiimejä ja kisaavat muita tiimejä vastaan. Sovelluksessa taukojumppaajat saavat pisteitä, etenevät tasoilla (kuva 1) ja tunnollisimmat voittavat rahan arvoisia palkintoja.



Kuva 1 Cuckoo Workout pistetasot (Rantalainen, 2015)

Sovelluksessa on hyvin onnistuttu rakentamaan terveellisempään elämäntapaan tähtäävästä toiminnasta leikillistä ja yhteisöllistä puuhastelua, johon on helppo saada mukaan koko työyhteisö.

3 FACEBOOK GRAPH API OSANA SOVELLUSTA

Useimmat sosiaaliset mediat tarjoavat erilaisia rajapintoja esimerkiksi käyttäjien tilitietoihin, datan analytiikkaan, markkinointiin tai sisällön jakamiseen. Voidakseen käyttää rajapintoja pitää kehittäjän rekisteröidä mobiilisovellus palveluntarjoajan kehittäjäpaneelissa. Esimerkiksi Googlen paneelissa voi aktivoida erilaisia API-palveluita pilvipalveluista karttoihin ja luoda API-avaimet, joilla Google yksilöi käyttäjän ja rajoittaa rajapintapyyntöjä. Ilmaiseksi voi tehdä vain rajoitetun määrän pyyntöjä päivässä, minkä takia tulee sovelluksen käyttöasteen kasvaessa ottaa käyttöön maksullinen versio. Muiden palveluntarjoajien järjestelmät toimivat samalla periaatteella. Kehittäjäportaaleissa voi säätää sovelluksen tietoturva-asetuksia ja API-avaimilla yksilöidään rajapintapyyntöjen lähde ainakin palvelinten välisissä pyynnöissä.

Subway Surfers on hyvä esimerkki sosiaalisten medioiden rajapintojen käytöstä. Kyseessä on mobiilipeli, jossa pelihahmo surffailee rautateillä ja metrotunneleissa väistellen vastaantulevia kulkuneuvoja. Peliin on upotettu mahdollisuus kirjautua sisään Facebookiin, minkä jälkeen kaverien pisteet tulevat näkyviin pistelistalle (kuva 2). Sovelluksesta on myös mahdollista tehdä julkaisu käyttäjän omalle Facebook-aikajanalle ja houkutella siten tuttaviaan kokeilemaan peliä. Näin sovelluksessa on yhdistetty kilpailu ja ilmainen mainonta.

Kuva 2 Subway Surfers pistetaulukko (Subway Surfers Leaderboard, 2013)

Toimeksiantoon kehitettäviä ominaisuuksia miettiessä syntyi ajatus hyödyntää sovelluksessa sosiaalisten medioiden rajapintoja. Tarjolla olevat vaihtoehdot rajattiin Facebookiin ja Googleen, sillä toimeksiantosovelluksessa oli jo ennestään olemassa sisäänkirjautumiset näille palveluntarjoajille. Valinta osui lopulta Facebookin Graph API -rajapintaan, sillä Facebookin käyttäjämäärät ovat aivan eri luokassa verrattuna Googlen sosiaalisen median palveluun.

3.1 Facebook Graph API

Graph API on HTTP-pohjainen rajapinta, jolla voi ohjelmallisesti pyytää dataa, tehdä julkaisuja, hallita mainoksia, ladata kuvia ja paljon muuta. Useimpiin pyyntöihin pitää liittää mukaan käyttäjäkohtainen avainkoodi (eng. access token), jonka Facebook luo sisäänkirjautumisen yhteydessä. Sisäänkirjautuminen pitää siis upottaa sovellukseen, jotta rajapinnasta saa mitään olennaista ulos. Käyttäjältä voi myös joutua pyytämään valtuutuksia riippuen tarvittavasta datasta: esimerkiksi sähköpostiosoitetta ei voi pyytää rajapinnasta ilman käyttäjän suostumusta.

Itse datapyynnöt rajapintaan ovat suhteellisen selkeitä. Yksinkertaisimmillaan yksittäinen pyyntö koostuu URL:sta, jonka perään lisätään pyydettävä data.

```
GET graph.facebook.com
  /me
```

Kuva 3 me-request (Facebook for developers, 2017b)

Esimerkiksi kuvan 3 ”/me”-pääte palauttaa käyttäjän nimen ja Facebook-tunnuksen. Kuvassa 4 on pyynnöstä palautuva JSON-muotoinen data.


```
{
  "name": "Testi Kayttaja",
  "id": "262228024289534"
}
```


Kuva 4 "me" - pyynnöstä palautuva JSON-data

Esimerkiksi profiilikuvan ja kaverilistan saa päätteellä ”/me?fields=picture,friends”. Facebook palauttaa kaverien tiedot sovelluksen käyttöön vain, jos sekä käyttäjä että hänen kaverinsa ovat antaneet luvat tietojen luovuttamiseen ja ovat sovelluksen käyttäjiä. Kuvassa 5 esimerkkikäyttäjän kaverit eivät täytä ehtoja, jonka takia kaverien dataosiossa palautuu tyhjää. Muuten datassa olisi tällä pyynnöllä kaverin nimi, Facebook-id ja profiilikuva.

```
{
  "picture": {
    "data": {
      "is_silhouette": false,
      "url": "https://scontent.xx.fbcdn.net/v/t1.0-1/p50x50/167
ee8691668aac52a714f1&oe=5A228910"
    }
  },
  "friends": {
    "data": [
    ],
    "summary": {
      "total_count": 4
    }
  },
  "id": "262228024289534"
}
```


Kuva 5 ”me?fields=picture,friends” – pyynnöstä palautuva data

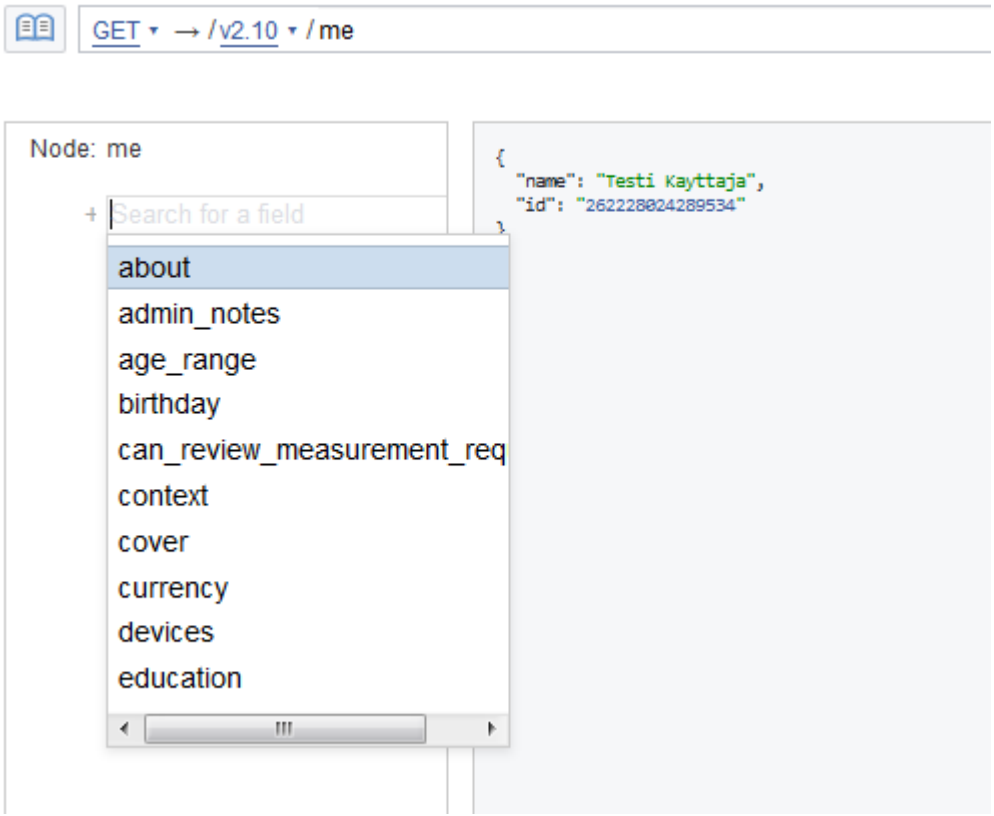
Rajapinnasta palautuvan datan muotoilua voi muokata jonkin verran. Pyydettävän datan perään voi laittaa lisäohjeita datan käsittelyyn. Esimerkiksi kuvan kommentit voi pyytää käänteisessä aikajärjestyksessä (kuva 6).

```
GET graph.facebook.com
  /{photo-id}?
    fields=comments.order(reverse_chronological)
```

Kuva 6 pyyntö kuvien kommentteihin käänteisessä aikajärjestyksessä (Facebook for developers, 2017d)

Facebookilla on palvelu Graph API Explorer, jolla voi rakennella erilaisia pyyntöjä Graph APIin. Palvelu tarjoaa ehdotuksia ja mahdollisia vaihtoehtoja ja tarkennuksia jo syötettyihin datasetteihin. Esimerkiksi /me-pyynnölle palvelu tarjoaa vaihto-

ehtoina mm. syntymäaikaa, valuuttaa, koulutusta ja ”yhteyksiä”, eli yhteisiä tuttavia ja tykkäyksiä (kuva 7).



Kuva 7 Graph API Explorer

Kuten sovelluksissa, myös palveluun pitää kirjautua avainkoodia varten.

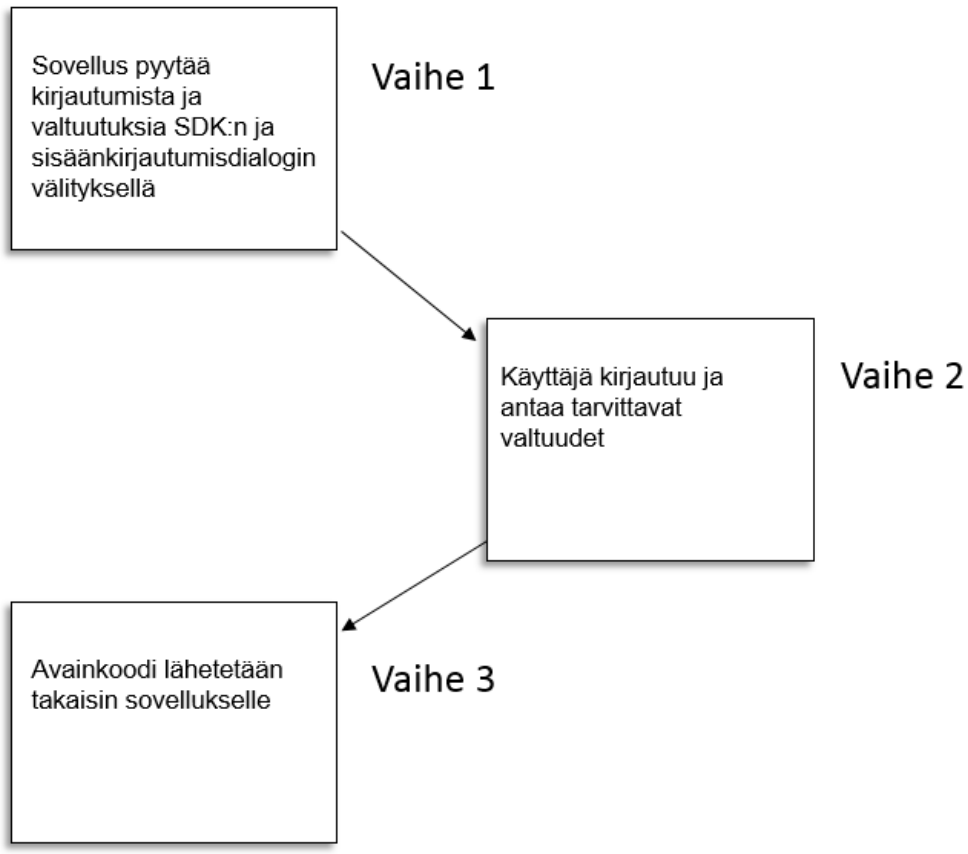
3.2 Facebook-sisäänkirjautumisen liittäminen sovellukseen

Facebook-sisäänkirjautuminen kolmannen osapuolen sovelluksissa on yksinkertaisuudessaan vain sovellukseen upotettu palikka, johon käyttäjä voi kirjoittaa tunnuksensa. Facebook tarjoaa valmiita eri ohjelmointikielille rakennettuja SDK-paketteja (eng. Software Development Kit), joilla toiminnallisuuden voi helposti integroida sovellukseen. Lisäksi on olemassa lukuisia kolmannen osapuolten rakentamia kehyksiä niille kielille, joille Facebook ei tarjoa omaa ratkaisuaan. Sisäänkirjautuminen on myös mahdollista rakentaa kokonaan itse.

SDK-pakettien käyttäminen on suhteellisen helppoa, mutta ne toimivat yleensä client-puolella. Joskus on kuitenkin tarpeen siirtää osa kirjautumisen jälkeisestä

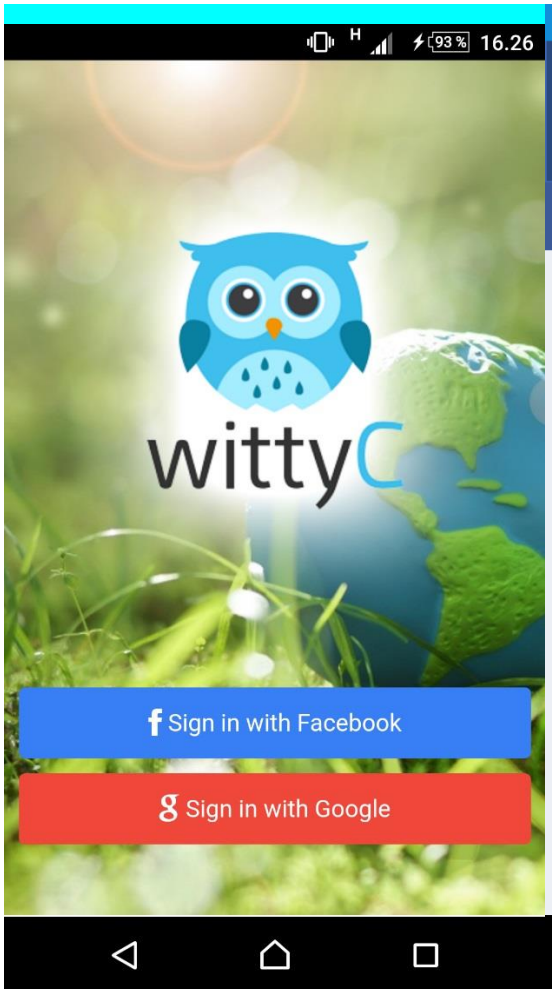
toiminnallisuudesta tietoturvan tai tietokantarakenteen takia palvelimelle. Silloin olisi hyvä ymmärtää, mitä kirjautumisprosessissa oikeastaan tapahtuu.

Kuvassa 8 on sisäänkirjautumisprosessi havainnollistettu kaaviolla.



Kuva 8 Facebook - kirjautumisprosessi

Vaiheessa 1 sisäänkirjautumisprosessi alkaa kuvan 9 nappia painamalla, minkä jälkeen avautuu kuvan 10 kirjautumisdialogi.



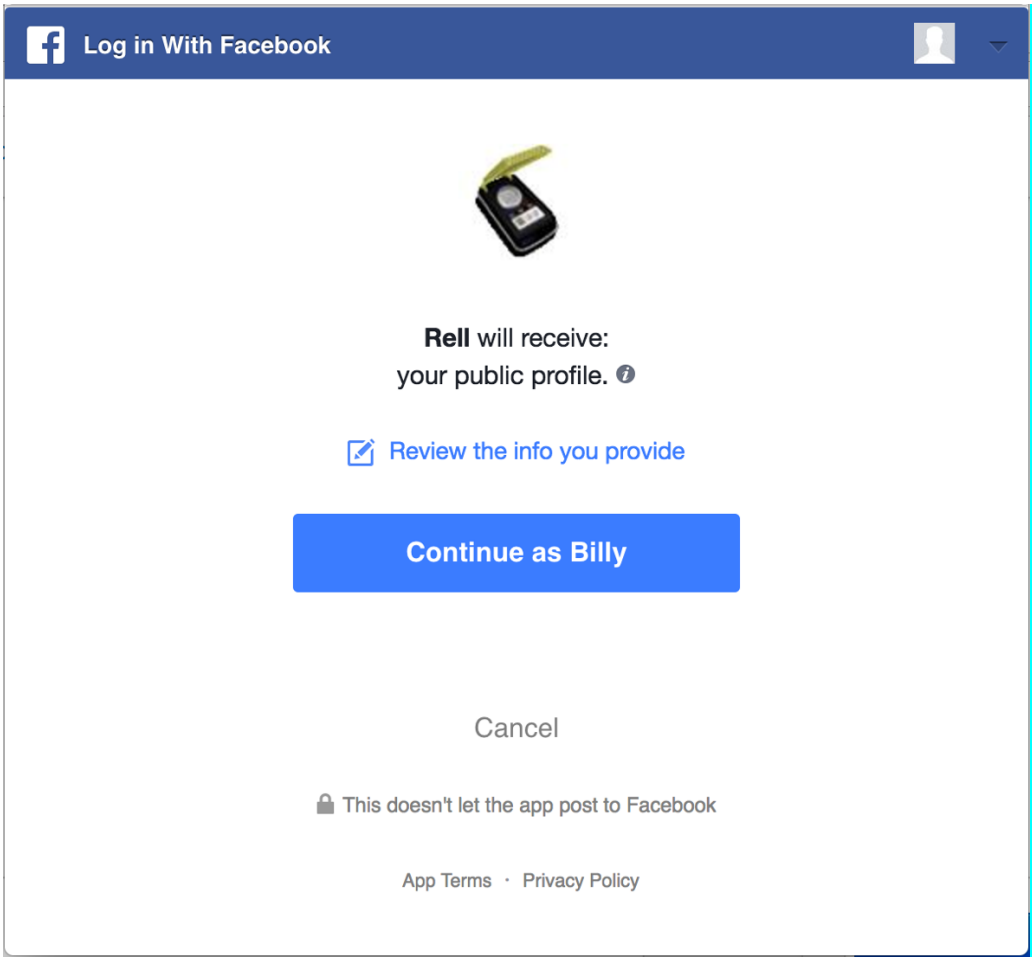
Kuva 9 Esimerkki sisäänkirjautumisnapista

Kuva 10 Napista avautuu sisäänkirjautumislomake

Perinteisesti ruudulle avautuva dialogi on voinut olla hybridisovelluksissa In-app browser eli sovelluksen sisällä toimiva verkkoselain, mutta vuoden 2017 alusta lähtien Google on voimakkaasti pyrkinyt korvaamaan kyseistä käytäntöä tietosuojahuolien takia erilaisilla käyttöjärjestelmien natiiviratkaisuilla. Oletettavaa on, että muut palveluntarjoajat seuraavat myöhemmin Googlen jalanjäljissä, jolloin nykyiset sovellukset joutunevat uusimaan kokonaan client-puolella olevat sisäänkirjautumisjärjestelmät.

Sisäänkirjautumisen yhteydessä käyttäjältä voidaan pyytää sovelluksen tarvitsemia valtuutuksia (kuva 11). Kaikkia valtuutuksia ei kuitenkaan tarvitse pyytää kerralla, vaan niitä voi pyytää myöhemmin sitä mukaa kun niille tulee tarvetta. Sovellus on luotettavampi ja käyttäjäystävällisempi, kun käyttäjä näkee konkreettisesti mihin kutakin pyydettävää lupaa tarvitaan sen sijaan, että kerralla pyydetään

kaikki mahdolliset luvat. Kun kyse on arkaluontoisista tiedoista, kannattaa sovelluksesta kehittää mahdollisimman luotettava ja läpinäkyvä.



Kuva 11 Sisäänkirjautumisnäkymä tunnistautumisen jälkeen (Facebook for developers, 2017c)

Vaiheessa 2 käyttäjä voi kirjautumisen jälkeen joko hyväksyä tai hylätä eri valtuutuksia yksitellen joko ensi kertaa kirjautuessa tai Facebookin kautta myöhemmin. Siksi sovelluksen kehityksessä tulee huomioida valtuutuksien muuttuminen lennossa.

Vaiheessa 3 sisäänkirjautumisprosessin onnistuttua Facebook lähettää takaisin sovellukselle käyttäjään sidotun lyhytikäisen avainkoodin. Niin kauan kuin avainkoodi on voimassa käyttäjän ei tarvitse kirjautua uudelleen API-pyyntöjä varten. Avainkoodin voi uusia ohjaamalla käyttäjä uudelleen kirjautumisprosessiin, jolloin jo sisään kirjautunut käyttäjä voi edetä prosessista läpi nopeasti tarvitsematta kirjoittaa tunnuksia uudelleen.

Jos sovellus käyttää selainpohjaista uudelleenohjauksiin perustuvaa itserakennettua kirjautumisprosessia, tulee ohjelman tarkistaa palvelinpuolella käyttäjän identiteetti ja tietojen oikeellisuus ennen suurempia jatkotoimenpiteitä. Selainpohjaisissa ratkaisuissa tiedot liikkuvat usein osoiterivillä, jolloin ulkopuolisten on mahdollista liittää osoiteriville omaa dataa oikean datan perään. Riippuen siitä, onko kirjautumisdialogissa valittu halutuksi vastaustyypiksi code vai token, voidaan palvelimella tarkistaa käyttäjätietojen oikeellisuus eri tavoilla. Code voidaan vaihtaa avainkoodiin pyynnöllä OAuth-pintaan (kuva 12).

```
GET https://graph.facebook.com/v2.10/oauth/access_token?
   client_id={app-id}
   &redirect_uri={redirect-uri}
   &client_secret={app-secret}
   &code={code-parameter}
```

Kuva 12 Code vaihto avainkoodiin (Facebook for developers, 2017c)

Vaihdon onnistuessa pyynnöstä palautuu JSON-merkkijono, jossa on sisällä Graph-API –pyyntöihin tarvittava avainkoodi (kuva 13). Vaihdon epäonnistuessa taas rajapinnasta palautuu virhettä selittävä viesti.

```
{
  "access_token": {access-token},
  "token_type": {type},
  "expires_in":  {seconds-til-expiration}
}
```

Kuva 13 Vastaus onnistuneesta vaihdosta (Facebook for developers, 2017c)

Token taas voidaan tarkistaa lähettämällä Facebookille kuvan 14 pyynnöllä tutkittava Token ja sovelluksen oma App access token tai sovelluskehittäjän avainkoodi. App access token voidaan luoda server-to-server -pyynnöllä käyttäen aikaisemmin sovittua salaista merkkijonoa.

```
GET graph.facebook.com/debug_token?
     input_token={token-to-inspect}
     &access_token={app-token-or-admin-token}
```

Kuva 14 Tokenin tarkistuspyyntö (Facebook for developers, 2017c)

Pyynnöstä palautuu vastaukseksi esim. kuvan 15 JSON-dataa tutkittavasta Tokenista.

```
{
    "data": {
        "app_id": 138483919580948,
        "application": "Social Cafe",
        "expires_at": 1352419328,
        "is_valid": true,
        "issued_at": 1347235328,
        "metadata": {
            "sso": "iphone-safari"
        },
        "scopes": [
            "email",
            "publish_actions"
        ],
        "user_id": 1207059
    }
}
```

Kuva 15 Vastaus Tokenin tarkistuspyyntöön (Facebook for developers, 2017c)

App_id ja user_id -auttavat sovellusta varmistamaan, että avainkoodi on pätevä sekä käyttäjälle että sovellukselle.

3.3 Facebook-avainkoodit

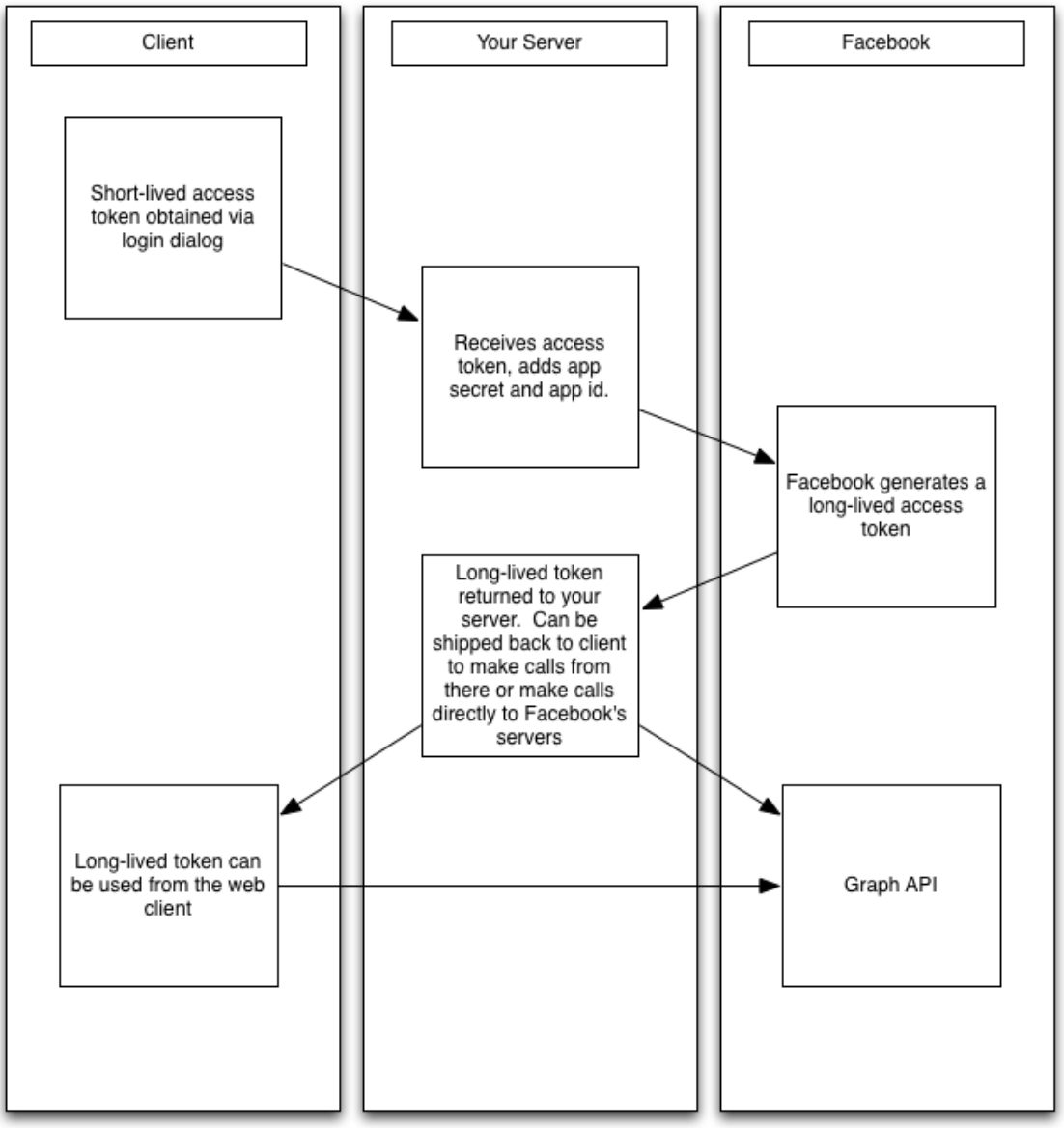
Käyttäjäavainkoodeja on kahdenlaisia: lyhyt- ja pitkäikäinen. Lyhytikäinen avainkoodi on voimassa vain muutaman tunnin kirjautumisen jälkeen ja pitkäikäinen yleensä noin 60 päivää.

Koska lyhytikäinen avainkoodi vanhenee melko nopeasti, riippuen sovelluksen käyttötapauksista, voi olla tarpeen pyytää Facebookilta pitkäikäinen avainkoodi. Se onnistuu vaihtamalla lyhytikäinen avainkoodi kuvan 16 API-pyynnöllä pitkäikäiseen.

```
GET /oauth/access_token?
    grant_type=fb_exchange_token&
    client_id={app-id}&
    client_secret={app-secret}&
    fb_exchange_token={short-lived-token}
```

Kuva 16 Lyhytikäisen avainkoodin vaihtopyyntö pitkäikäiseen (Facebook for developers, 2017a)

Kuvassa fb_exchange_token on lyhytikäinen avainkoodi, app-id sovelluksen Facebookiin rekisteröinnin yhteydessä luotu identiteettitunnus ja app-secret salassa pidettävä avainkoodi, jolla todennetaan sovelluksen palvelimelta tulevat API-pyynnöt. App-secret -avainta ei tule ikinä laittaa mihinkään asiakkaalle annettavaan ohjelman koodiin, koska sillä voi muokata sovellusta Facebookin konsolissa. Koska App-secret -avain tarvitaan pyyntöön mukaan, sitä ei tule koskaan tehdä client-puolelta. Siksi sovellukselle tarvitaan oma palvelin, joka lähettää pyynnön rajapintaan. Kuvassa 17 on havainnollistettuna client-puolen, sovelluksen palvelimen ja Facebook-rajapinnan välinen kommunikointi.



Kuva 17 Avainkoodin vaihtoprosessi (Facebook for developers, 2017a)

1. Client-puolella käyttäjä kirjautuu sisään ja hänelle luotu lyhytikäinen avainkoodi lähetetään palvelimelle.

2. Palvelin muodostaa pyynnön, lisää siihen tarvittavat avainkoodit ja lähettää sen Facebookin rajapintaan.

3. Facebook käsittelee pyynnön ja vaihtaa lyhytikäisen avainkoodin pitkäikäiseen.

4. Uusi avainkoodi palautuu takaisin palvelimelle, josta se voidaan joko lähettää edelleen client-puolelle tai tehdä suoraan palvelimelta uusia pyyntöjä Facebookin rajapintaan.

On tärkeää huomata, että vanhentunutta avainkoodia ei voida käyttää vaihdossa uuteen pitkäikäiseen. Lyhytikäisen avainkoodin voi uusia ohjaamalla käyttäjä uudelleen sisäänkirjautumisprosessiin.

4 CASE WITTYC

Toimeksiannon kohteena on johdannossa esitelty WittyC. Se on hiilijalanjäljen laskemiseen ja tarkkailuun kehitetty mobiilisovellus, johon kehitin osana toimeksiantoa erilaisia sosiaalisia toiminnallisuuksia. Rakensin sovellukseen Facebook Graph APIa hyödyntäen kaverilistan, sovelluksen sisäisen sisällönjakojärjestelmän sekä mahdollistin Google- ja Facebook-sisäänkirjautumisten yhdistämisen. Esittelen tässä luvussa sovelluksessa käytettävät tekniikat ja kehittämäni toiminnallisuudet.

4.1 Käytettävät tekniikat

WittyC:ssä on käytössä useita erilaisia ohjelmointitekniikoita. Client-puoli on rakennettu AngularJS:n ja Ionicframeworkin yhdistelmällä. Palvelinpuolella pääasiallinen ohjelmointikieli on Java. Ohjelmointia helpottamassa ja palvelinpuolen rakennetta yksinkertaistamassa on Javalle rakennettu kirjasto Hibernate.

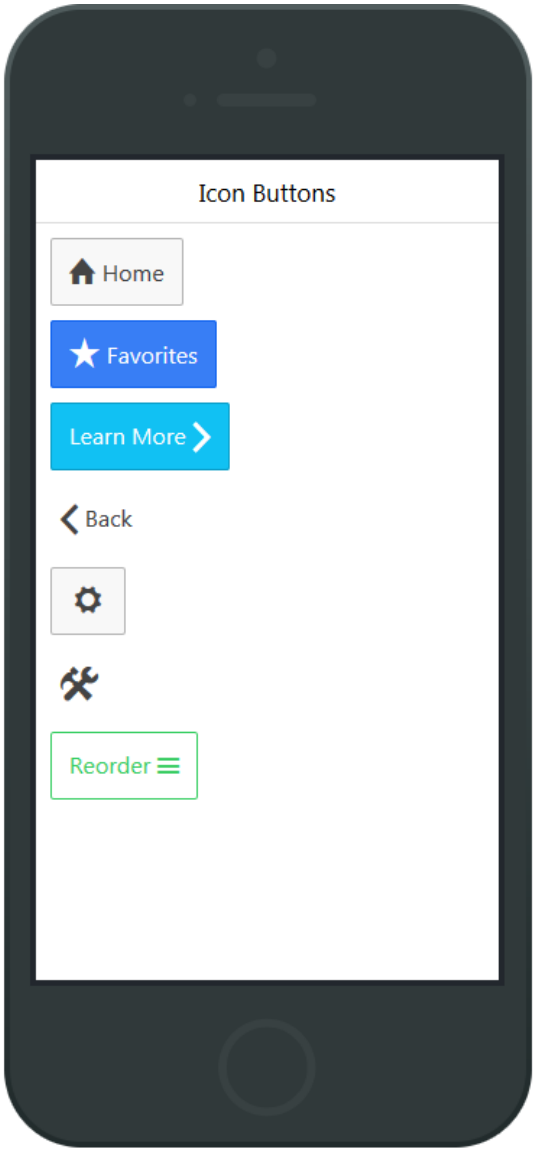
AngularJS ja Ionicframework

AngularJS on Javascriptille rakennettu MVC-mallinen (Model-View-Controller) koodikirjasto, jossa on pyritty yksinkertaistamaan eri osien välistä toimintaa. AngularJS toimii sitomalla model ja view toisiinsa siten, että kun yksi muuttuu, myös toinen päivittyy. Esimerkiksi tekstikentästä voidaan tehdä model ”yourName” ja sitoa se H1 otsikkoelementtiin, jonka tekstinä on ”Hello {{yourName}}”. Tekstikenttään kirjoitettu nimi päivittyy reaaliajassa otsikkoon (kuva 18).

```
<!doctype html>
<html ng-app>
  <head>
    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.6/
angular.min.js"></script>
  </head>
  <body>
    <div>
      <label>Name:</label>
      <input type="text" ng-model="yourName" placeholder="Enter a name here">
      <hr>
      <h1>Hello {{yourName}}!</h1>
    </div>
  </body>
</html>
```

Kuva 18 AngularJS Data Binding esimerkki (AngularJS, 2017)

Ionicframework on HTML5 SDK (Software Development Kit) tai UI-kirjasto (User Interface), jonka tarkoituksena on helpottaa sovellusten front endin, eli asiakkaille näkyvien osien rakentamista tarvitsematta miettiä erilaisten laitealustojen omia vaatimuksia. Ionicissä on oma sisäänrakennettu järjestelmä erilaisten elementtien kuten nappien, valikoiden, ikonien ja vastaavien luomiseen. Elementit skaalautuvat automaattisesti näytön leveyden mukaan ja tuntuvat ja näyttävät kuin natiivisovellukselta. Esimerkkinä Ionicin ulkoasusta kuvassa 19 on erilaisia Ionic-nappeja, joiden väriä, kokoa, ulkoasua, tekstejä tai vaikkapa ikonien sijaintia voi muuttaa.



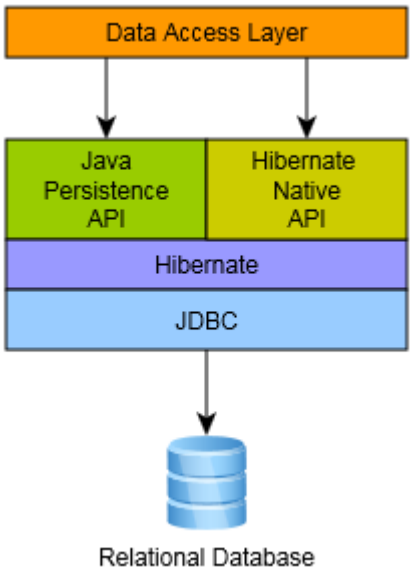
Kuva 19 Erilaisia Ionic-nappeja (Ionic Documentation, 2017)

Ionic on jo alusta asti suunniteltu toimimaan saumattomasti yhteen AngularJS:n kanssa, minkä vuoksi kehittämisvaiheessa ei pitäisi tulla yhteensopimattomuuden takia onglemia Ionicin ja AngularJS:n välillä.

Hibernate

Hibernate ORM (Object/Relational Mapping) on Java-ympäristöille rakennettu ratkaisu, jonka tehtävänä on mahdollistaa objektimallin mukaan esitetyn datan muunnos relaatiotietokantamalliin ja päinvastoin. Hibernate ei siis huolehdi vain

muunnoksista Java-luokista tietokantaan (ja Java-datatyypeistä SQL-datatyyppeihin), vaan tarjoaa myös datan pyyntö- ja hakupalveluita. Se voi huomattavasti vähentää kehitysaikaa, joka muuten käytettäisiin datan manuaaliseen käsittelyyn SQL:ssä ja JDBC:ssä. (Hibernate ORM 5.2.12 Final User Guide, 2017). Kuvassa 20 on yleiskatsaus Hibernaten arkkitehtuuriin.



Kuva 20 Hibernaten rooli palvelinpuolella (Hibernate ORM 5.2.12 Final User Guide, 2017)

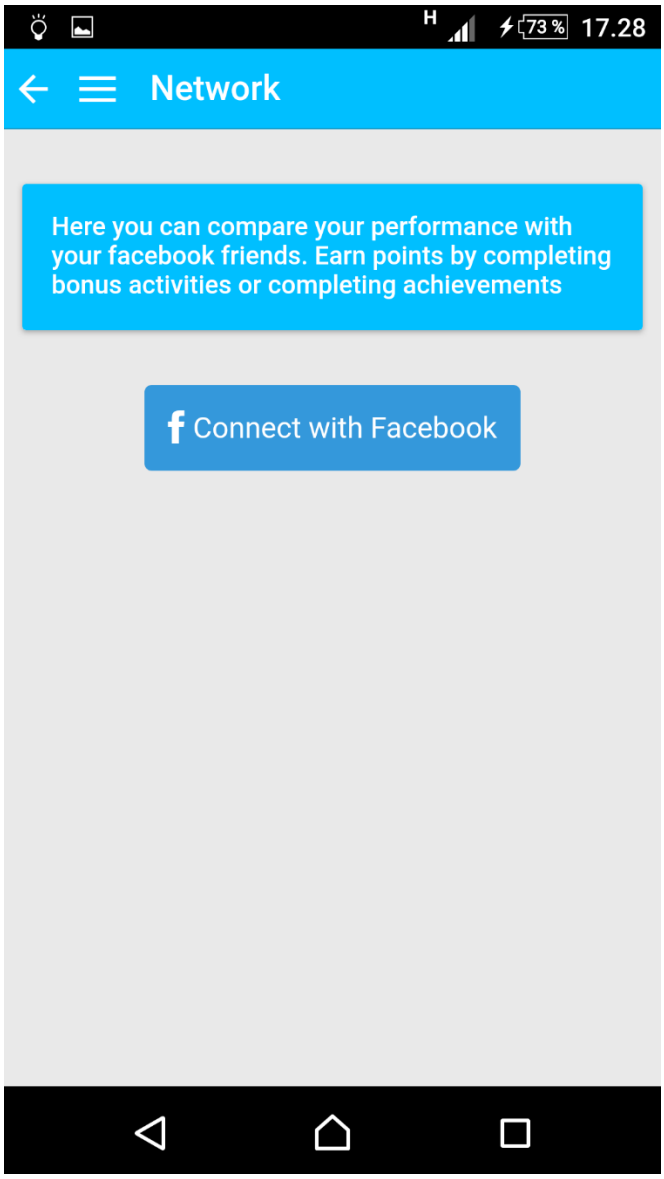
ORM-ratkaisuna Hibernate ”istuu” Java-sovelluksen datan käyttökerroksen ja relaatiotietokannan välissä. Java-sovellus käyttää Hibernaten rajapintoja datan lataamiseen, tallentamiseen, kyselyyn yms. (Hibernate ORM 5.2.12 Final User Guide, 2017).

JDBC (Java Database Connectivity) on Javan rajapinta, joka tarjoaa standardin Java-ohjelmointikielen ja erilaisten tietokantojen välille (Oracle Technology Network, 2017).

4.2 Google-kirjautumisen yhdistäminen Facebookiin

WittyC-sovellukseen voi kirjautua sisään Googlella tai Facebookilla. Sovelluksen yleiskäytön kannalta kirjautumistavalla ei ole merkitystä, mutta rakentamani sosiaaliset toiminnallisuudet hyödyntävät Facebookin rajapintoja, eivätkä siten toimi

ilman Facebook-tunnuksia. Jotta käyttäjä saa sovelluksesta sen kaiken potentiaalin irti, oli tarpeen lisätä sovellukseen mahdollisuus yhdistää Google-kirjautujan profiiliin Facebook-tunnukset. Tämä onnistui yksinkertaisimmillaan lisäämällä Facebook-sisäänkirjautuminen sinne, missä ominaisuutta tarvitaan (kuva 21).



Kuva 21 Google-kirjautuja ei ole vielä yhdistänyt Facebook-tiliä

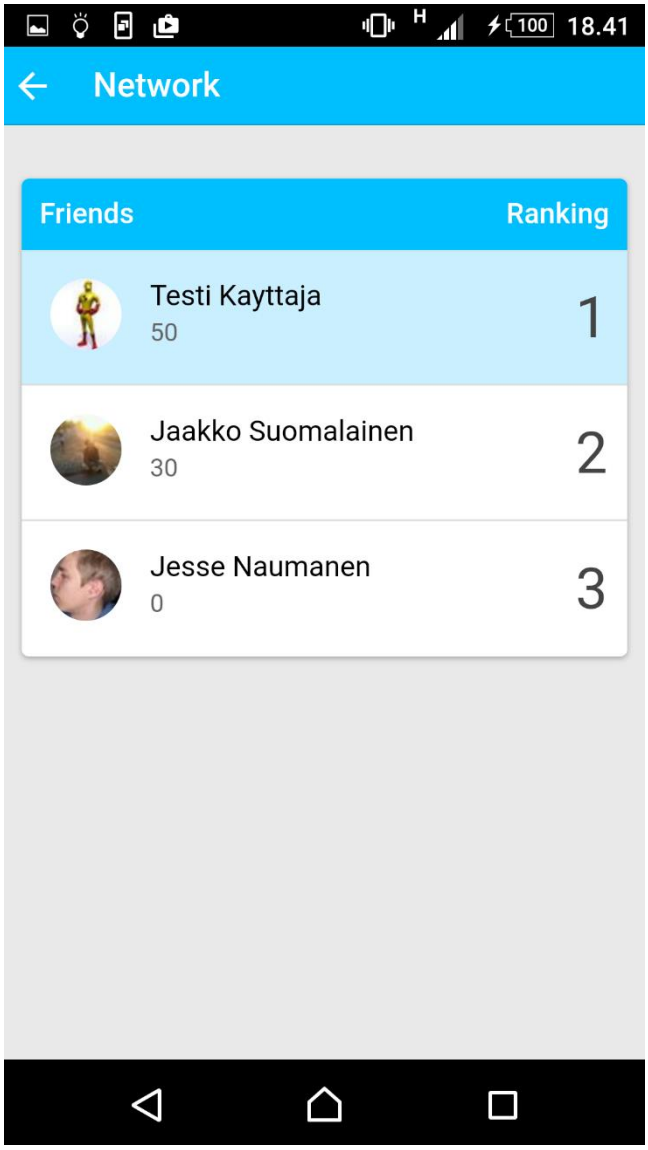
”Connect with Facebook” -nappia painettaessa avautuu käyttäjälle samanlainen kirjautumisdialogi kuin kuvassa 10. Käyttäjä voi kirjautua Facebook-tunnuksillaan sisään, minkä jälkeen käyttäjän Facebook ID voidaan tallentaa tietokantaan. Tässä vaiheessa on huomioitava Facebook-avainkoodin tallennustapa, sillä tunnuksien yhdistämisen jälkeen käyttäjä voi kirjautua omalle tililleen molemmilla

tunnuksilla. Kirjautuessa sisään Google-tunnuksilla sovellus ei luonnollisesti saa Facebookilta uutta avainkoodia, mikäli aikaisempi koodi on vanhentunut tai ei käytettävissä. Eri tallennuspaikoilla on omat ongelmansa, minkä takia se kannattaa harkita tarkkaan. Esimerkiksi avainkoodin tallennus tietokantaan on ongelmallista siten, että avainkoodi saattaa vanhentua, muuttua tai muuten poistua käytöstä aivan yllättäen. Miksi siis tallentaa tietokantaan dataa, joka ei välttämättä ole seuraavana päivänä validia? Toinen vaihtoehto avainkoodin säilytyspaikaksi voisi olla localStorage. Valitettavasti käyttäjä pystyy sen halutessaan tyhjentämään ja useampi käyttäjä samalla laitteella saattaa olla ongelmallista. Useamman avainkoodin tallentaminen localStorageen ja niiden yhdistäminen oikeaan käyttäjään on huomattavan monimutkaista isossa sovellusjärjestelmässä. Yksinkertaisin ja helpoin ratkaisu on siten tallentaa avainkoodi localStorageen ja tyhjentää se uloskirjautumisen yhteydessä. Ratkaisu ei kuitenkaan ole ihanteellinen Google-kirjautujille, sillä käytännössä sosiaalisien ominaisuuksien käyttäminen vaatii Facebook-kirjautumiseen siirtymistä myös yhdistämisen jälkeen.

4.3 Kaverilista

Osana WittyC:n kehittämistä sosiaalisempaan suuntaan rakensin sovellukseen kaverilistan. Se on perusta kaikenlaisille toiminnallisuuksille, jotka hyödyntävät käyttäjän kaveriverkostoja. Ominaisuuden käyttämisen tarvitaan Facebook-tunnuksia, sillä listan kaverit haetaan suoraan Facebookin Graph APIsta.

Datan hakemiseen käytetään käyttäjältä saatua avainkoodia. Tämä avainkoodi lähetetään sovelluksen palvelimelle, joka vuorostaan lähettää pyynnön Facebookille. Mikäli Graph APIsta palautuu haluttu kaveridata, lisää palvelinpuoli kaverilistaan mukaan käyttäjien pisteet. Näin käyttäjät voivat vertailla kaveripiiriensä tuloksia pistetaulussa (kuva 22).



Kuva 22 Kaverilista

Valitettavasti kolmannen osapuolen datan hyödyntämisessä on aina sovelluksen kehittämisen kannalta ongelmia. Oikeanlaisen datan vastaanottamisen varmistaminen vaatii suuren määrän erilaisia tarkistuksia (kuva 23).

```
if (vm.authType === 1 || ((vm.user.facebookId !== null && vm.user.facebookId !== "") && vm.
    if (vm.accessToken !== null){
        console.log(vm.accessToken);

        if (typeof vm.accessToken !== null) {

            $ionicLoading.show({
            template: "<ion-spinner class='spinner-stable' icon='android'></ion-spinner></b
                $filter('translate')('SPINNER_LOADING_FRIENDS')
            });


            UserService.getFacebookFriends(vm.accessToken).then(
            function (result) {
                console.log(result);
                if(typeof result === "undefined") {
                    $ionicLoading.hide();
                    vm.errorMessage = $filter('translate')('SOMETHING_WENT_WRONG');
                } else if (typeof result === "object"){

                    if ("friends" in result){
                        vm.friendlist = result.friends.data;
                        vm.getScores();
                    }

                    else if ("error" in result) {

                        if (result.error.code === 190) {
                            $ionicPopup.alert({
                                title: $filter('translate')('FACEBOOK_TOKEN_EXPIRED'),
                                template: $filter('translate')('PLEASE_LOGIN_AGAIN_RENEW')
                            });
                        } else {

                            $ionicPopup.alert({
                                title: $filter('translate')('SOMETHING_WENT_WRONG'),
                                template: $filter('translate')('CANT_FETCH_FRIENDS')
                            });
```

Kuva 23 Tarkistuksia Graph APIsta palautuvalle datalle. Kuvassa vain pieni osa erilaisista tarkistuksista.

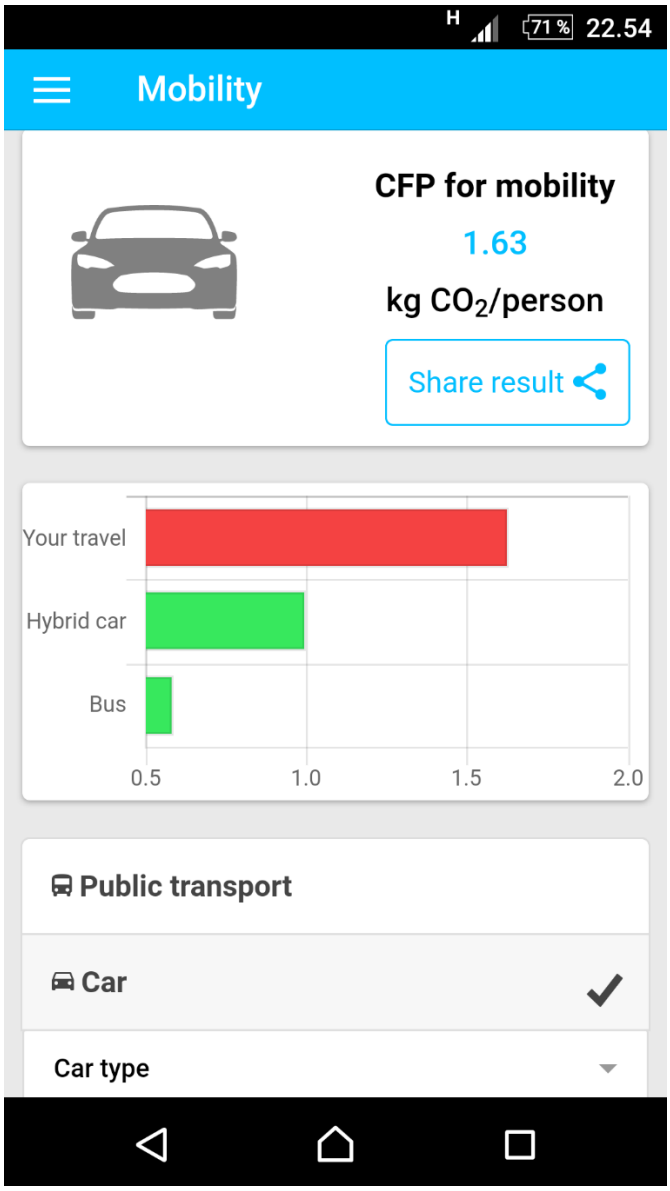
Esimerkiksi Facebookin palvelimet voivat olla saavuttamattomissa, käyttäjän avainkoodi voi olla vanhentunut, käyttäjä on voinut perua erilaisia valtuutuksia tai vaikkapa vastaanotettu data on vääränlaista. Moni asia voi mennä pieleen kolmen eri tahon välisessä tiedonsiirtorakenteessa ja käyttäjälle pitäisi saada lähetettyä ongelmaa erittelevä virheviesti. Kun otetaan asetelmaan mukaan erilaiset puhelinmallit ja useista eri paikoista kytkettävät luvat ja valtuutukset, niin soppa on valmis.

4.4 Saavutusten ja laskujen jakaminen

Tulosten ja saavutuksien jakaminen on tärkeä osa ihmisten välistä kanssakäymistä. Mahdollisuus kertoa läheisimmillemme hyvistä tuloksista on keskeinen osa

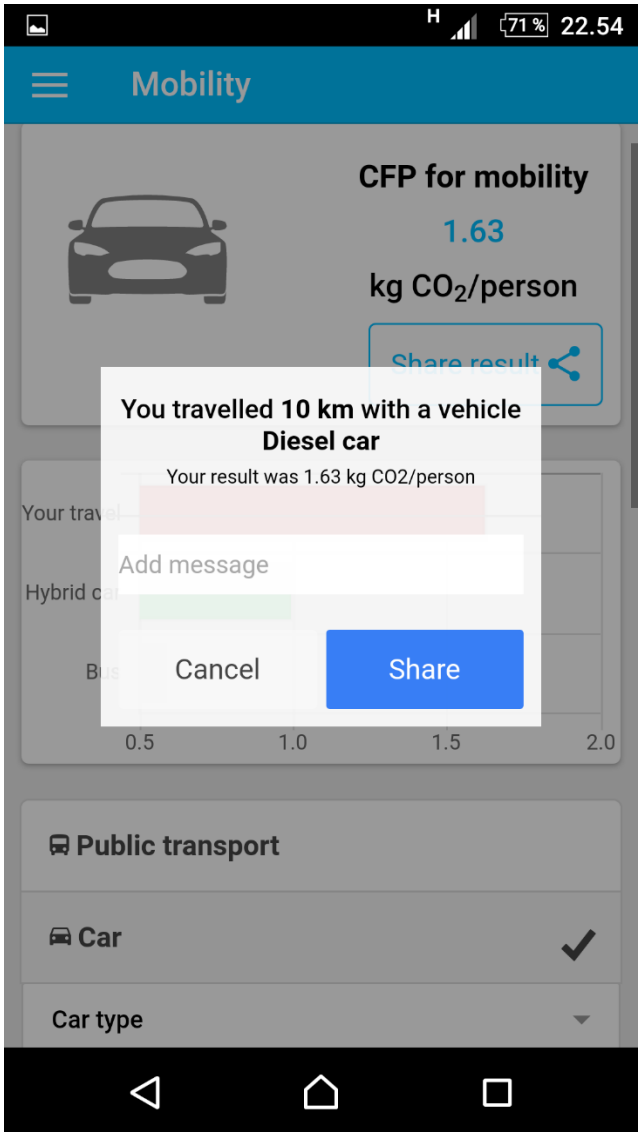
useita nykypäivän sovelluksia. Sovellukseen rakentamani jakotoiminnallisuuden tavoitteena oli mahdollistaa käyttäjien välinen tulosten, saavutusten ja laskuhistorian jakaminen.

Kuvassa 24 jakonappi tulee näkyviin suoritetun laskun jälkeen.



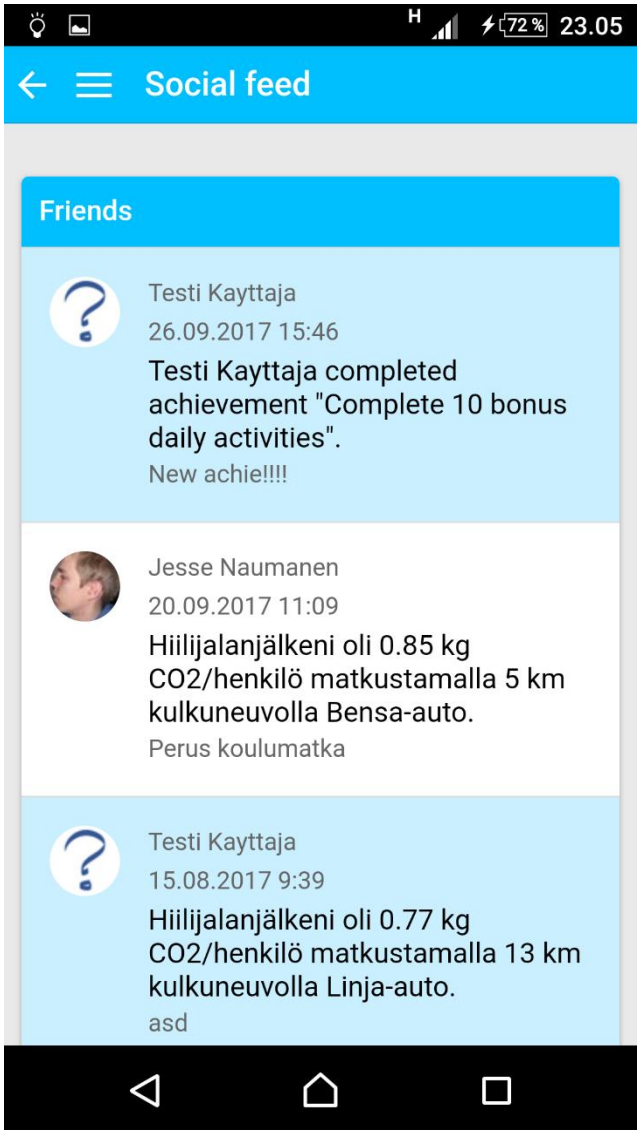
Kuva 24 Laskun tulosten jakonappi

Napista painaessa avautuu Ionic-popup-dialogi, johon voi kirjoittaa viestin tulosten mukaan (kuva 25). Osa tekstistä luodaan automaattisesti, jossa tyypillisesti kerrotaan laskun tulokset ja mistä osiosta se on peräisin.



Kuva 25 "Share result" -napista avautuu Ionic popup -ikkuna

Jakamiset tulevat näkyviin omalle sivulleen. Kuvassa 26 yksittäisessä jaossa on vasemmalla profiilikuva, ylhäällä jakaja ja jakoaika, keskellä automaattiteksti ja alhaalla jakajan oma viesti.



Kuva 26 Tehdyt jaot

Nykyisellään jakojen kieli riippuu jakajan kieliasetuksista. Jos sovelluksen kieliasetukseen on valittu suomi, myös jakojen automaattiteksti luodaan suomeksi. Tuloksena jakolistauksessa voi olla tekstejä useilla kielillä.

5 PÄÄTÄNTÖ

Opinnäytetyöni tavoitteena oli tutkia sosiaalisten toiminnallisuuksien hyödyntämistä mobiilisovelluksissa, esitellä sosiaalisten medioiden rajapintoja ja toimeksiantona kehittää WittyC-sovellukseen sosiaalisia toiminnallisuuksia osana pelillistämisprojektia. Vaikka lähteiden löytäminen sosiaalisista toiminnallisuuksista oli

melko hankalaa, löytyi tietoa lopulta pelillistämistä käsittelevistä teoksista ja dokumenteista. Pureuduin myös Facebookin sisäänkirjautumis- ja Graph API rajapintoihin, joita sovelsin toimeksiannossa. Toimeksiantoa varten kehitin WittyC-sovellukseen kaverilistan, laskujen ja saavutusten jakojärjestelmän sekä yhdistin Facebook- ja Google-sisäänkirjautumisen. Toimeksiantaja hyväksyi työni tulokset ja sovellus julkistettiin suljetusti Google Play Storessa betatestaajille.

Toimeksiannon tekeminen oli melko pitkä ja työläs prosessi. Vaikka kehitettävät ominaisuudet olivat näennäisesti melko yksinkertaisia, vaati niiden rakentaminen huomattavia määriä ohjelmointia kulissien takana. Suurin osa työstä koostui palvelinpuolen ohjelmoinnista ja Facebookin datan käsittelystä. Olen kuitenkin melko tyytyväinen lopputulokseen, sillä pääsin tavoitteisiin ja kehitin toimeksiantoon pyydetyt toiminnallisuudet.

Työtä voisi kehittää eteenpäin esimerkiksi jakamisen osalta. Esimerkiksi mahdollisuus jakaa tuloksia suoraan Facebookiin voisi olla hyvä lisäominaisuus. Rakentamani sovelluksen sisäistä jako- ja kaverijärjestelmää voisi kehittää toimimaan ilman Facebook-sisäänkirjautumista tai ottaa mukaan useampia sosiaalisia medioita. Myös jakojen valmistekstien sisältöjä ja jakojärjestelmän rakennetta voisi parantaa. Tekstejä on sekä teknisesti että sisällöllisesti vaikea kääntää eri kielille eivätkä ne nykyisellään tuo merkittävästi lisäarvoa viesteihin.

Opinnäytetyö oli minulle jatkumoa WittyC-projektissa. Olen ollut enemmän tai vähemmän mukana sovelluksen kehityksessä sen alusta lähtien. Tästä kehitysprosessista on ollut minulle valtavasti hyötyä kokemusten ja oppien muodossa. Onnistumisien ja virheiden myötä olen saanut selkeämmän kuvan mobiilisovellusten kehityksestä sekä itsestäni ohjelmoijana.

LÄHTEET

AngularJS. 2017. The Basics. WWW-dokumentti. Saatavissa: https://angularjs.org/ [viitattu 26.11.2017].

Facebook for developers. 2017a. Expiration and Extension of Access Tokens. WWW-dokumentti. Saatavissa: https://developers.facebook.com/docs/facebook-login/access-tokens/expiration-and-extension [viitattu 30.8.2017].

Facebook for developers. 2017b. Graph API Overview. WWW-dokumentti. Saatavissa: https://developers.facebook.com/docs/graph-api/overview [viitattu 25.9.2017].

Facebook for developers. 2017c. Manually Build a Login Flow. WWW-dokumentti. Saatavissa: https://developers.facebook.com/docs/facebook-login/manually-build-a-login-flow [viitattu 15.9.2017].

Facebook for developers. 2017d. Using the Graph API. WWW-dokumentti. Saatavissa: https://developers.facebook.com/docs/graph-api/using-graph-api/ [viitattu 25.9.2017].

Groh, F. 2012. Gamification: State of the Art Definition and Utilization. Ulm University. Institute of media informatics. PDF-dokumentti. Päivitetty: 4.9.2016. Saatavissa: http://hubscher.org/roland/courses/hf765/readings/Groh_2012.pdf [viitattu 2.4.2017].

Haonperä, J. 2013. Mitä on pelillistäminen? Blogi. Saatavissa: http://www.cloudriven.fi/blogi/mita-on-pelillistaminen [viitattu 2.4.2017].

Ionic Documentation. 2017. Icon Buttons. WWW-dokumentti. Saatavissa: https://ionicframework.com/docs/v1/components/#icon-buttons [viitattu 26.11.2017].

Mihalcea, V., Ebersole, S., Boriero, A., Morling, G., Badner, G., Cranford, C., Bernard, E., Grinovero, S., Meyer, B., Ferentschik, H., King, G., Bauer, C., Andersen, M. R., Maesen, K., Vansa, R., Jacomet, L. 2017. Hibernate ORM 5.2.12 Final User Guide. WWW-dokumentti. Saatavissa: http://docs.jboss.org/hibernate/orm/5.2/userguide/html_single/Hibernate_User_Guide.html [viitattu 26.11.2017].

Oracle Technology Network. 2017. Java SE Technologies – Database. WWW-dokumentti. Saatavissa: http://www.oracle.com/technetwork/java/javase/jdbc/index.html [viitattu 26.11.2017].

Rantalainen Yhtiöt. 2015. Cuckoo Workout's workout challenge. WWW-dokumentti. Saatavissa: http://www.rantalainen.fi/en/clients/cuckoo-workout/ [viitattu 8.11.2017].

Schell, J. 2008. The Art of Game Design - A Book of Lenses. Burlington: Morgan Kaufman Publishers.

Subway Surfers Leaderboard. 2013. Post your best score. Facebook-tilapäivitys. Saatavissa: https://www.facebook.com/Subway-Surfers-Leaderboard-280264835438559/ [viitattu 8.11.2017].